International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 7 | ISSN : 2456-3307

Comparative Study of Apache Hadoop vs Spark

Varsha KR
Department of Computer Science and Engineering, RV College of Engineering Bangalore, Karnataka, India

ABSTRACT

The paper focuses on analyzing the differences and comparative study of two most talked about frameworks – Hadoop and Spark – both of which have increasing potential for the big data management. The analysis is carried out regarding components, design, data storage, recovery from failure among other features. Comparative analysis is made by executing certain algorithms on two platforms and comparing the execution time. Similarly, suitability of frameworks for different scenarios is discussed.

Keywords : Hadoop, Spark, Map-Reduce, HDFS

I. INTRODUCTION

Spark is being shown to be 100x faster compared to Map-Reduce of Hadoop. But, originally, Spark was created with the intention of being used as an extension of the existing Hadoop framework. Hadoop is an environment or an ecosystem in which processing on big data can be done whereas Spark is an application which provides an interface to process big data. Spark requires a file system like the HDFS to store data. The objective of the paper to find differences between Hadoop and Spark is between two computing paradigms Hadoop MapReduce and Spark.

Definitions

As defined by Apache Foundation-

Hadoop - The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.

Spark - Apache Spark is a fast and general engine that enables users to run large-scale data analytics applications across clustered systems.

MapReduce – MapReduce is a procure employed by Hadoop where data processing happens in two stages – Map and reduce. Map phase involves distributing the dataset across various nodes where computation/ processing is performed independently for each divided portion of the data. Reduce phase involves combining results obtained in all nodes using a corresponding 'Join'/reduce operation.

II. BASIC COMPONENTS

The basic components making up these two frameworks gives the basic difference between the two.

Hadoop is composed of four modules:

a) Hadoop Common - libraries and utilities

b) Hadoop Distributed File System (HDFS) - a distributed file system

c) Hadoop YARN - a resource-management system

d) Hadoop MapReduce - large scale data processing, and computing framework

Spark is composed of :

a) Cluster Manager - Standalone, Hadoop Yarn, Apache Mesos
b) Distributed File system - HDFS
c) Computing framework - Spark itself is the computing framework and unlike Hadoop is not an ecosystem.
d) Library - Spark SQL, Spark Streaming, MLLib, Graph X

III. COMPARISION OF FEATURES

A. Implementation and Infrastructure

In 2014, there was a contest Daytona Gray Sort measuring how fast a system can sort 100 TB of data (1 trillion records). Spark used 206 AWS EC2 machines and sorted 100 TB of data on disk in a meager time of 23 minutes. The previous record was held by MapReduce, it used 2100 machines and took 72 minutes. Spark did the same thing as MapReduce, but only 3 times faster on 10 times fewer hardware.

Product	Hadoop	Spark
Implementation	Ecosystem	Compute engine
Resource Management	Hadoop YARN	Standalone, YARN, Mesos
Library	Hadoop Common	MLLib, GraphX, Streaming, SparkSQL
File system	HDFS	HDFS, Cassandra, Amazon S3
Computing framework	MapReduce	Spark
Execution unit	Process	Thread
Data model	Key, Value (Java object)	Key, Value (RDD)
Intermediate data handling	Disc/Local (and network)	Spark collectives(network)
Language	Java	Scala (runs on java VM)

B. Design

The design of these two frameworks is analyzed in two phases: Map and Reduce. The differences can be noted in the table as follows:

Hadoop	Spark
Map Phase	
Each Map task produces (key, Value) pairs which is stored in Circular buffer (around 100MB)	The output is written to OS Buffer cache.
Data is spilled to disk when buffer is 80% full	OS decides when data is spilled to disk.
On a particular node, many map tasks are run and many spill files are created. Hadoop merges all the spill files, on a node, into one big file which is sorted and partitioned based on number of reducers.	Each map task creates as many shuffle spill files as number of reducers. SPARK doesn't merge and partition shuffle spill files into one big file. the map tasks which run on the same cores will be consolidated into a single file.
Reduce phase	
Pushes the intermediate/spill files to memory	Pulls spill files to reduce side
If buffer reaches 70%, data is spilled to disk	Data written to memory directly
Spills are merged	If data doesn't fit OOM exception is thrown (before 0.9)
Reduce method called	Reducer method called

C. Data Storage model

Data in Hadoop is distributed across several data nodes. Files are split into blocks each of 64 or 128 MB. These blocks are replicated (standard 3 nodes). Name node stores metadata i.e. details about blocks making up a file and location of blocks.

Spark integrates with many systems like HDFS (explained above), Cassandra, HBase etc. The main abstraction Spark provides is a resilient distributed dataset (RDD). RDDs are a 'immutable resilient distributed collection of records' which can be stored in the volatile memory or in a persistent storage. All processing of data happens on RDDs and transforming them. All data flow happens in-memory except at t2 where there is insufficient space so it is stored on disk.

Whereas in case of MapReduce, all data flow happens through disk and at each step data is stored on disk with copious amounts of reading and writing.

D. Recovery from Failure

In case of Hadoop, if a node fails, then the task Tracker on that node stops sending heartbeats to Job Tracker (frequency 3s). The JT which checks the heartbeat every 200s declared the node to be dead if no heartbeat is received for 600s. The data that is stored on that node is recreated from the other two nodes having the same data. In Spark internally records every transformation that was applied to build the RDD into a Direct Acyclic Graph (DAG) called lineage. On failure of any part, it is recreated following the DAG data flow. Recovery happens via Write ahead logs if the system fails during an operation. Here, the intention of the operation is first written down into a durable log, and then the operation is applied to the data. On failure, system can recover by reading the log and reapplying the operations it had intended to do.

E. Spark without Hadoop

Spark doesn't require Hadoop to run. If we are not reading the data from HDFS, Spark can run on its own. There are many other storages such as S3, Cassandra, etc., from which Spark can read and write data. Under this architecture, Spark runs in stand-alone mode not requiring Hadoop components in any way.

IV. ANALYSIS OF EXECUTION OF PROGRAMS

A. Machine Learning Algorithms

Hadoop Map Reduce seems to be inefficient to run applications which reuse a working set of data repeatedly as in cases of iterative algorithms and interactive data mining.

The spark framework takes an edge off with efficient implementation of machine learning procedures for most ML algorithms run on the same data set iteratively and in MapReduce, there is no effortless way to communicate a shared state from iteration to iteration. Some attempts to run ML on Hadoop and Spark and their comparison was done by UC, Berkeley with page rank, logistic regression and k-means clustering. The obtained results strongly proved that Spark with its MLLib is an excellent framework for ML algorithms and the its efficiency is far greater than the traditional procedure done on Hadoop.

The fundamental idea here is that of the fine grained mutable state offered by Hadoop is a very low-level abstraction. Here, with the writable interface associated with <key, value> pairs the datatype needs to be mutable to participate in the serialization/de-serialization process. This design was done, perhaps to reduce the amount of garbage objects. Mutable means that there are updates, so it's possible for different replicas of a piece of data to become inconsistent. But on the other hand, Spark's coarse grained immutable data as RDDs offer a higher level

of abstraction as one operation is applied to an entire dataset saving the number of operations or functions.

The following algorithms were run both on Hadoop and Spark and the results obtained as follows:

B. Interactive Analytics

MapReduce initially supported only two operations-map and reduce. But interactive analytics requires a greater number of steps. There is also a non-availability of communication between shared states. There exist solutions to these like cascading or the usage of elevated level SQL languages. But these too involve multiple steps and thus cannot achieve the required latency.

On the contrary, Spark allows caching of data and operates on in-memory RDDs. There exists a common location of shared data which can be used interactively till the end of the session. Hence interactive analytics can be achieved at extensively greater speeds compared to many step data extractions from data-nodes in MapReduce.

C. Batch processing and stream processing

Hadoop's MapReduce is a batch processing framework on Hadoop ecosystem. Spark is also a batch processing framework originally but it includes a library which can be used for streaming called Spark Streaming. Batch processing is very efficient in processing high volume data. Where data is collected, entered to the system, processed and then results are produced in batches. In contrast, stream processing involves continual input and outcome of data. It emphasizes on the Velocity of the data. Data must be processed within small time period or near real time Streaming processing gives decision makers the ability to adjust to contingencies based on events and trends developing in real-time

D. Real Time Processing

Real time processing involves the ability of the system to respond and react to real time stimuli. Data must be processed fast so that the enterprise using the system can estimate changing conditions and take decisions accordingly. It has application in Network monitoring, intelligence and surveillance, risk management, e-commerce, fraud detection, smart order routing, transaction cost analysis, pricing and analytics, market data management, algorithmic trading, data warehouse augmentation among others.

Real time processing requires system to be operational 24X7 and efficient recovery from failures. As Spark streaming is built on Spark with RDD abstraction and a feature to have write ahead logs(journal) exhibits a potential recovery mechanism. Yahoo uses Spark for personalizing news pages for web visitors and for running analytics for advertising. Conviva uses Spark Streaming to learn network conditions in real time. Hadoop like streaming is not a powerful system for real-time processes. It uses Apache Flume for data streaming and Apache Storm for real time, event stream processing.

V. OTHER PARAMETERS FOR COMPARISION

A. Use cases

There are a few cases in which Hadoop MapReduce out performs, those which do not involve much of

communication and iterations. The biggest advantage that Hadoop MapReduce may have is if the data size is bigger than memory, then Spark can't leverage the cache and might be even slower than MapReduce's batch processing with more disk hits. For instance, ETL type computations where result sets are large and may exceed aggregate RAM of the cluster by an order of magnitude.

B. Security

In case of Hadoop, it uses Kerberos SPNEGO for security. By default, Hadoop is in non-secure mode. To run in secure mode, kinit commands of Kerberos is used. The filter then delegates to Authentication handler, obtains authentication token and sets a signed cookie. For clients, the signed cookie and its validity is verified, information is extracted and then sent to target. Spark supports authentication via a shared secret. It sends request to Authentication server through SSH whereas Hadoop MapReduce communications are configured to use HTTPS. A difference with Spark is that each subsequent request to the API must include a token and be properly signed.

VI. CONCLUSION

Spark overtakes Hadoop when it comes to performance and ease of coding but it sure cannot replace it completely as there are many use cases of Hadoop MapReduce owing to its security and ability of batch processing.

VII. REFERENCES

1. Verma Ankush, Mansuri Ashik Hussain, Jain Neelesh, "Big Data Management Processing with Hadoop MapReduce and Spark Technology: A Comparison", 2016 Symposium on Colossal Data Analysis and Networking (CDAN).
2. S Humbetov, "Data-intensive computing with map-reduce and hadoop", International Conference on Applcation of Information and Communication Technologies, pp. 1-5, 17-190ct. 2012.
3. Polato Ivanilton, R Reginaldo, Goldman Alfredo, Kon Fabio, "A comprehensive view of Hadoop research-A systematic literature review", Journal of Network and Computer Applications, vol. 46, pp. 1-25, November 2014.
4. online] Available: http://spark.apache.hadoop.org/.
5. online] Available: http://spark.apache.org/docs/latest/mllib-Iinear-methods.html.